VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Absolvovanie individuálnej odbornej praxe

Individual professional practice in the company

2017 Dušan Paušly

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Dušan Paušly**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Absolvování individuální odborné praxe
Individual Professional Practice in the Company

Jazyk vypracování: čeština

Zásady pro vypracování:

1. Student vykoná individuální praxi ve firmě: KVADOS, a.s.
2. Struktura závěrečné zprávy:
   a) Popis odborného zaměření firmy, u které student vykonal odbornou praxi a popis pracovního zařazení studenta.
   b) Seznam úkolů zadaných studentovi v průběhu odborné praxe s vyjádřením jejich časové náročnosti.
   c) Zvolený postup řešení zadaných úkolů.
   d) Teoretické a praktické znalosti a dovednosti získané v průběhu studia uplatněné studentem v průběhu odborné praxe.
   e) Znalosti či dovednosti scházející studentovi v průběhu odborné praxe.
   f) Dosažené výsledky v průběhu odborné praxe a její celkové zhodnocení.

Seznam doporučené odborné literatury:

Podle pokynů konzultanta, který vede odbornou praxi studenta.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Pavel Krömer, Ph.D.**

Konzultant bakalářské práce: Ing. Antonín Vaněček

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne. Uviedol som všetky literárne zdroje a publikácie, z ktorých som čerpal.

V Ostrave 18. júna 2017 .....

Souhlasím se zveřejněním této bakalářské/diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v bakalářských/magisterských programech VŠB-TU Ostrava.

V Ostrave 18. júna 2017 .....

Rád by som v prvom rade poďakoval Ing. Antonínovi Vaněčkovi za umožnenie absolvovania bakalárskej praxe. Tiež by som rád poďakoval doc. Ing. Pavlovi Krömerovi za rady pri vypracovaní bakalárskej práce a celému vývojovému týmu spoločnosti Kvados, a.s. za odbornú pomoc, cenné rady a konzultácie.

Abstrakt

Táto práca popisuje priebeh bakalárskej praxe v spoločnosti KVADOS, a.s. Prax som vykonával od 2. septembra 2016 do 20. mája 2017. V prvej časti práce poukazujem na základné informácie o spoločnosti, jej zameraní a pôsobení. Ďalej popisujem úlohy, ktorým som sa venoval počas praxe. Na konci práce som zhrnul získané znalosti štúdia na VŠB-TUO, ktoré som uplatnil pri praxi a novo nadobudnuté znalosti vďaka absolvovaní danej praxe.

Kľúčové slová: KVADOS, a.s., Bakalárska prax, Angular 2, TypeScript, HTML, CSS, XML, .NET, C#, Multiplatformná webová aplikácia, Single-page application, Metadáta, CQRS, T4 Template, Framework, Modul, web-klient, myTEAM NG, QAS

Abstract

This thesis describes the process of my bachelor's professional practice in the company KVADOS, a.s. I was performing practice from 2. September 2016 to 20. May 2017. In the first part I' m focusing on based informations about company and its operation. Then I mention the tasks, that I have been involved in during practice. By the end of this thesis, I described the knowledge, that I gained while studying at the VŠB-TUO, which I applied in practice and knowledge acquired in practice and evaluated the contribution of the whole bachelor's professional practice.

Key Words: KVADOS, a.s., Bachelor's practice, Angular 2, TypeScript, HTML, CSS, XML, .NET, C#, Multiplatform web client, Single-page application, Metadata, CQRS, T4 Template, Framework, Modul, web-client, myTEAM NG, QAS

Obsah

Seznam použitých zkratek a symbolů

CQRS	–	Command Query Responsibility Segregation
SPA	–	Single-page application
XML	–	eXtensible Markup Language
HTML	–	Hyper Text Markup Language
DOM	–	Document Object Model
CSS	–	Cascading Style Sheets

Zoznam obrázkov

Zoznam tabuliek

Zoznam výpisov zdrojového kódu

1 Úvod

Získavanie praktických znalostí a využitie teórie v praxi je dôležitou časťou vzdelávania. Práve toto bolo dôvodom, prečo som sa pred necelým rokom rozhodol pracovať popri štúdiu. Podarilo sa mi získať stáž v spoločnosti KVADOS a.s., kde som nastúpil na pozíciu vývojára v oddelení výskumu a vývoja. Nasledujúci školský rok som sa rozhodol absolvovať bakalársku prax ako bakalársku prácu v spolupráci s KVADOS a.s. V prvej časti opíšem spoločnosť KVADOS a moje pracovné zameranie. V druhej časti stručne zhrniem využité technológie na projekte. V poslednej sa zameriam na úlohy, ktoré mi boli pridelené. K vypracovaným úlohám priložím postup riešenia, ukážky kódu a obrázky jednotlivých navrhnutých komponent.
Na konci práce zhodnotím získané vedomosti, využité znalosti získané počas štúdia na VŠB-TUO a moje celkové pôsobenie v spoločnosti počas praxe.

2 Popis praxe v spoločnosti KVADOS, a.s.

2.1 O spoločnosti KVADOS

Spoločnosť KVADOS a.s. sa zaoberá vývojom a dodávateľstvom vlastných softwarových riešení, služieb a poradenstva. Spoločnosť pôsobí a sídli v Ostrave od roku 1992 a zameriava sa prevažne na klientelu zo segmentu obchodu a služieb. Je poskytovateľom softwaru, ktorého cieľom je zvýšiť kvalitu riadenia procesov. Pôsobí na českom i zahraničnom trhu a aktuálne sa softwarové riešenia spoločnosti uplatňujú v dvanástich krajinách sveta. Ako jedna z mála spoločností na českom trhu je KVADOS a.s. certifikovanou firmou podľa medzinárodných noriem ISO [6]. Tiež patrí k zakladajúcim členom združenia IT Cluster, ktorého súčasťou je aj Fakulta elektrotechniky a informatiky VŠB-TU Ostrava.

Obr. 1: Logo Kvados, a.s.

2.2 Popis pracovného zamerania

Náplň mojej bakalárskej práce odpovedala požiadavkám spoločnosti, aj náplni práce vývojového týmu. Tento tým pozostáva z desiatich členov a pomaly sa rozrastá. Od začiatku praxe som spolupracoval na pokrokovej, multiplatformnej aplikácií myTEAM NG s využitím platformy .NET a Angular 2 frameworku.

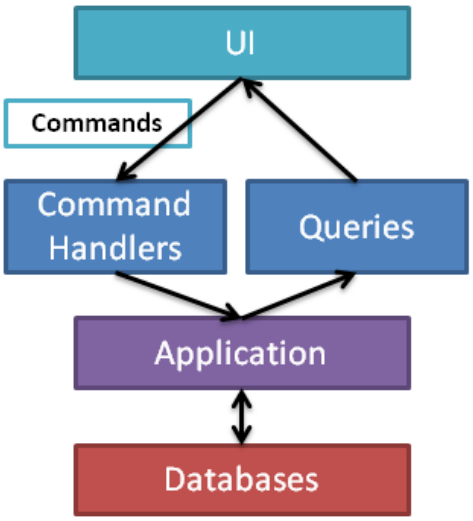
Prešiel som úlohami od vývoja nových komponent, cez výrobu panelov, po základné testovanie a opravu chýb pred nasadením u klienta. Spočiatku som dostával jednoduché úlohy kvôli zoznámeniu sa s technológiou AngularJS a programovacím jazykom TypeScript. Neskôr mi boli priradené pokročilejšie, na tvorbu formulárových komponent a písanie vlastných Extenzií, DataView a ActionCommandov.

3 Technológie a postupy využité pri bakalárskej praxi

3.1 Command Query Responsibility Segregation

CQRS je návrhový vzor, ktorý sa zaoberá prácou so škálovateľnými informáčnými systémami [5]. Pre prácu s informačnými systémami sa väčšinou používajú CRUD operácie. Tie nám umožňujú čítať, zapisovať, zaktualizovať a zmazať dáta. Na rozdiel od CRUD operácií, CQRS pracuje s dátami iným spôsobom. Tento návrhový vzor je vhodný predovšetkým vtedy, keď nám CRUD operácie nestačia a potrebujeme zahrnúť ďalšie operácie. Je vhodné ho použiť pri dvoch typoch aplikácií, ktoré ale nebývajú príliš časté. Sú to veľmi komplexné aplikácie, pri ktorých sa s CQRS môžeme lepšie vysporiadať s komplexitou a veľmi veľké alebo vyťažené aplikácie, ktoré potrebujú škálovať.

CQRS oddeluje zápis a čítanie informácií. Zápis a editácia informácií sú spostredkované v systéme pomocou príkazov, v našom prípade takzvaných ActionCommands (Commands) a čítanie informácií prostredníctvom dotazov – Dataviews (Query). Implementácia príkazov a dotazov je odlišná [5]. Operáciu čítania môžeme nahradiť viacerými dotazmi a poddotazmi. V praxi to znamená vytvorenie dotazu, ktorý vráti celý zoznam záznamov a dotaz, ktorý vráti jeden záznam z informačného systému. Samozrejme, základným pravidlom je, že ActionCommand vykonáva výhradne zmenu, nič nevracia a zároveň Dataview nemení stav dát, len ich číta.



Obr. 2: Model návrhového vzoru CQRS

3.2 Single-page application

SPA je webová aplikácia, ktorá je umiestnená na jednej webovej stránke. V SPA je všetok potrebý kód (HTML, JavaScript, CSS) riešený jednorázovým načítaním alebo sú príslušné zdroje

dynamicky načítané a pridané na stránku podľa potreby. Stránka sa v žiadnom bode opätovne nenačíta ani neposkytuje prechod na inú stránku [4].

3.3 Angular 2

AngularJS je voľne dostupný, webový Framework, postavený na jazyku Javascript. Zameriava sa na tvorbu Single-page aplikácií. Aplikácie sú tvorené za pomoci HTML kódu, do ktorých sú vložené formátujúce značky, určujúce, aké dáta sa majú na miesto vložiť – data-binding. Data-biding sa v kóde vyznačuje zloženými zátvorkami {{}}. Kód vo vnútri zátvoriek je výrazom, tým pádom v ňom môžeme volať funkcie vracajúce hodnoty a iné operácie [1].

Obr. 3: Logo AngularJS

Angular 2 je postavený v jazyku JavaScriptového typu s názvom TypeScript. Nie je úplne novým jazykom, je to rozšírenie ES6, ktorý je ďalšou verziou JavaScriptu po ES5. ES5, tiež známy ako „regular JavaScript“, je normálny bežne používaný JavaScript. TypeScript je výsledkom spolupráce medzi spoločnosťami Microsoft a Google. TypeScript priniesol päť základných vylepšení oproti ES5:

1. Typovosť
2. Triedy
3. Importy
4. Anotácia
5. Jazykové nástroje (template string, fat arrow function syntax)

Ďalšími dôvodmi využitia TypeScriptu/ES6 sú funkcie, ako napr.:

1. Rozhrania
2. Genericita
3. Import a Export modulov
4. Destructuring

Angular 2 aplikácia nie je nič viac ako strom komponent. Koreňom tohto stromu a zároveň komponentou s najvyššou úrovňou je aplikácia sama. To je to, čo bude prehliadač vykreslovat pri bootavaní (bootstrapping) aplikácie [1]. Jedna zo základných vecí ohľadom komponent je to, že sú skladateľné. Tým pádom môžeme veľké komponenty vyskladať z viacerých menších podkomponent. Jednoducho povedané, aplikácia je komponenta, ktorá volá iné komponenty.

Každá komponenta sa skladá z troch základných častí: Annotation, View a Controller. Anotáciu v programátorskom kóde nájdeme v tvare @Component. Tá pridáva metadáta do triedy, ktorá ju nasleduje. Je to miesto, kde sa konfiguruje komponenta. Predovšetkým ale bude konfigurovat spôsob interakcie okolitého sveta s komponentou. Annotation sa skladá zo selectoru a template. Selector určuje, ako je komponenta rozpoznaná v HTML šablóne.

```
<invoice-in-detail-block></invoice-in-detail-block>
```

Výpis 1: Selector v jazyku HTML

View je vizuálna časť. Použitím prvku template deklarujeme HTML šablónu a CSS štýly, ktoré bude komponenta obsahovať. Controller je definovaný triedou.

3.4 Metadáta

Metadáta sú dáta, ktoré nám poskytujú informáciu o iných dátach. Môžu obsahovať informáciu o vlastných dátach, ako aj informáciu o kontexte. Bývajú štrúktúrované za pomoci tagovanej reprezentácie. Štruktúra a formát metadát sú v aplikáciách dohodnuté a štandartizované. V našom prípade, pri tvorbe webovej aplikácie sú metadáta užitočné pre vyhľadávanie vo väčšom množstve informácií. Pridané metadáta ku stránke prinesú vyššiu organizačnú štruktúru. Stránka (HTML) obsahuje metadáta v hlavičke (header) a zapisujú sa pomocou tagu.

```
<meta></meta>
```

Výpis 2: Tag pre Metadáta v jazyku HTML

Do nich je možné vkladať informácie typu:

- Popis stránky
- Názov aplikácie

- Aktuálna verzia aplikácie
- Jazyk aplikácie
- Autor
- ...

3.5 T4 Template

Text Template Transformation Toolkit (T4) je šablónový, text generujúci framework od spoločnosti Microsoft, ktorý je súčasťou Visual Studio. Zdrojové kódy sú označované príponou .tt. Využíva sa ako súčasť aplikácie alebo nástroj na automatizáciu vytvárania textových súborov s rôznymi parametrami a akéhokoľvek textového formátu. T4 používa vlastný template formát, ktorý môže obsahovať .NET kód ako riadiacu logiku a textové bloky (v našom prípade HTML kódu) [3].

4 Zoznam pridelených úloh počas bakalárskej praxe

V tejto časti vypíšem úlohy, na ktorých som pracoval počas bakalárskej praxe. Dôkladnejšie sa v práci budem venovať dôležitejším a rozsiahlejším agendám a ich úlohám. Následná tabuľka obsahuje identifikátor úlohy, typ a názov.

ID	**Typ**	**Názov**
68606	Task	Objednávky vydané - Hlavička založení - OrderOutHeaderCreateWizardStep
68622	Task	Objednávky vydané - Založení - Summary - OrderOutSummaryWizardStep
68791	Task	WebClient - seznámení se s technologií
68958	Task	Číselníky - Evidence firem - CompanyListLookupDataView
69585	Task	Objednávky vydané - Položky (needitovatelné) - OrderOutItemList - Panel
69680	Task	Objednávky vydané - Možnosti - OrderOutOptions - Panel
69924	Bug	Oprava chyb - obecné
69943	Task	Faktury přijaté - Detail - InvoiceInDetailDataView
69995	Task	Faktury přijaté - Základní údaje - InvoiceInDetail - Panel
70446	Task	Podpora výroby
70581	Task	Faktury přijaté - Poznámka - InvoiceInNote - Panel
70651	Task	Odeslání objednávky e-mailem - OrderOutSendEmail
70806	Task	Odeslání objednávky e-mailem - OrderOutEmailAttachmentList
70807	Task	Odeslání objednávky e-mailem - Chybějící přílohy - OrderOutEmailAttachmentMissing
70816	Bug	Objednávky vydané - Zobrazení názvu stavu objednávky
70847	Task	WebClient - Podnadpisy - State/ControlData Extension
70849	Task	Odeslání objednávky e-mailem – OrderOutSendEmailActionCommandHandler
70992	Task	Faktury přijaté - Přílohy - Odstranění - InvoiceInAttachmentRemove - Panel
72153	Task	WebClient - ValidationErrorsBlock Component
72382	Task	Zarovnávání notifikačních e-mailů
72705	Task	Home Page - dlaždice
74535	Task	UI Refaktoring 2016 - Obecné
74538	Task	UI Refaktoring 2016 - ListMenu + Grid
74540	Task	UI Refaktoring 2016 - Základní údaje
74542	Task	UI Refaktoring 2016 - Rychlý přehled
74546	Task	UI Refaktoring 2016 - Formuláře - Nápověda
74548	Task	UI Refaktoring 2016 - Průvodce

Tabuľka 1: Prehľad pridelených úloh počas bakalárskej praxe

5 Zadané úlohy a postup riešenia

5.1 WebClient - seznámení se s technologií

Na začiatku môjho pôsobenia mi bola pridelená úloha, ktorá ma mala oboznámiť s technológiou Angular 2, SPA, CQRS, Workflow a využitím Metadát. Celkový čas strávený úlohou bol vyčíslený na 80 hodín.

Obsahom tejto úlohy bolo štúdium odborných materiálov, ktorými boli Ng-book 2 a videolekcie. Zahŕňal aj prípravu vývojového prostredia, stiahnutie TypeScript balíčkov, nainštalovanie programov ako Visual Studio 2015, Microsoft SQL Management Studio 2014 ale aj doplnkov, ako ReSharper. Podarilo sa mi úlohu splniť a do 40 hodín som začal pracovať na vývoji produktu.

5.2 Faktury přijaté - Základní údaje - InvoiceInDetail – Panel

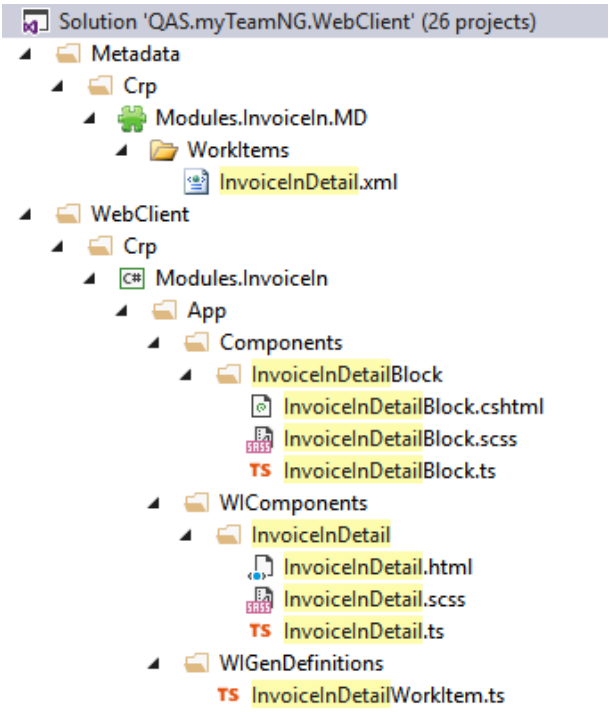
Prvá Úloha pridelená v rámci produktu bola na panel výpisu detailu založenej faktúry. Zadaný čas na vypracovanie bol 16 hodín. Po preštudovaní analýzy som zistil, aké dáta a grafické prvky (komponenty) bude detail obsahovať. Každý panel stránky je WorkItem v aplikácií. WorkItem pozostáva zo štyroch častí:

1. Definície v XML súbore
2. WorkItem komponenty obsahujúcej .html, .scss, TypeScript súbory
3. Volaných komponent
4. Vygenerovanej definície v TypeScripte

Ako prvý krok som zvolil vytvorenie XML definície. XML definícia sa skladá z viacerých častí: relácie, WorkItem/WorkItemPart, extenzie, View, dáta a príkazy. Nasledujúcim krokom bolo pridanie relácie. Za pomoci relácií sa predáva identifikátor, v našom prípade ID faktúry, medzi panelmi. WorkItem môže mať viacero relácií a predkov, v tomto prípade je predkom iba panel Rýchleho prehľadu faktúry. V panely detailu je nutné volať Extenziu na načítanie identifikátoru do netriviálnej funkcie (Dataview), aby vedelo, z ktorej faktúry má dáta zobraziť. Podľa analýzy, má podnadpis v hlavičke panelu zobraziť číslo dokladu faktúry s prefixom. Ten vypisujem za pomoci Extenzie, ktorá použije číslo dokladu z dotazu.

Časť View pozostáva z vyzobrazených komponent. V našom prípade volám komponentu vo formáte bloku, ktorého zložku s podsúbormi si musím ručne vytvoriť. V časti s dátami nadeklarujem Dataview a medzi príkazy pridám command, ktorý mení podnadpis podľa zmeny identifikátoru.

Komponenta obsahuje .cshtml, .scss a TypeScript súbory. TypeScript súbor obsahuje štruktúru podľa Angular 2. Vďaka predkovi môžeme použiť zdedenú funkcionalitu na výpis stavu faktúry. Súbor s kaskádovými štýlmi by mal obsahovať len importy striktne nadefinovaných fontov a farieb pre aplikáciu. V .cshtml súbore je telo detailu pozostávajúce z tried blokov, riadkov a



Obr. 4: Štruktúra zložiek WorkItemu

stĺpcov. Výpisané dáta sa získavajú z databáze podľa komplexných SQL dotazov nad viacerými tabuľkami.

WorkItem komponenta a jej štruktúra je vygenerovaná šablónou. Content .html súboru obsahuje selector na komponentu.

Pri úlohe som sa naučil vytvárať WorkItemy, pochopiť ich štruktúru, pracovať s Extenziami a data-bidingom. Precvičil som si prácu s HTML a CSS štýlmi.

5.3 Faktury přijaté - Detail – InvoiceInDetailDataView

Neoddeliteľnou úlohou pre detail faktúry je vytvorenie takzvaného DataView, ktorý je komplexným, netriviálnym dotazom na SQL databázu. Zadaný čas na vypracovanie úlohy bol 16 hodín.

Dáta, ktoré sa majú zobrazovať užívateľovi pozostávajú z dvanástich tabuliek a ich následnému prepojeniu. Posielať cez sieť všetky tieto dáta by bolo značne neefektívne, takže bolo potrebné vytvoriť komplexný dotaz, ktorý všetky výpočty a selekcie nad danými tabuľkami spracovával na strane databáze. Tým bol značne zrýchlený beh aplikácie a požiadavok klienta. Dotaz bude použiteľný vo viacerých paneloch aplikácie, z toho dôvodu musí byť modifikovateľný a rozšíriteľný. Prvým krokom bolo vytvorenie súboru dotazu v module faktúr. Pre upresnenie, každá agenda aplikácie má svoj vlastný modul. Je to z dôvodu lepšej údržby kódu, prehľadnosti

a hlavne požiadavku klienta. Nasledovala deklarácia atribútov ako použitých tabuliek. Implementácia dotazu obsahovala SQL klauzule písané v .NET kóde. V dotaze je zahrnutá aj validácia údajov v klauzule WHERE pre dodatočné overenie správnosti vložených údajov.

Kvôli bezpečnosti a korektnosti na strane web-klienta bolo nutné dáta získané z databáze prekonvertovať do požadovaného tvaru konštánt, lebo dáta s atribútom v tvare „tabulka.parameter“ nie sú vyhovujúce. Tieto dáta od zákazníka, získateľné z prehliadača, by mohli byť ľahko zneužité. O zmienenú funkcionalitu sa stará metóda v druhej časti DataView a prekonvertuje dáta do správneho formátu. Posledným krokom bola registrácia DataView do registru modulu faktúr. Kvôli prehľadnosti som do častí kódu zapracoval komentáre a metódy rozdelil do regiónov.

```
// Contact Person
string partnerSurname=dataItem.GetValue<string>(m_kParPerson.Table.Surname);
string partnerFistname=dataItem.GetValue<string>(m_kParPerson.Table.FirstName);
if (!string.IsNullOrEmpty(partnerFistname) && !string.IsNullOrEmpty(
    partnerSurname))
{
   dataBagObject.Add(InvoiceInDetailDataViewConstants.
       PartnerShortNameOutputItem, string.Format("{0} {1}.", partnerSurname.Trim
       (), partnerFistname.Length > 1 ? partnerFistname.Substring(0, 1) :
       partnerFistname));
}
else
{
   dataBagObject.Add(InvoiceInDetailDataViewConstants.
       PartnerShortNameOutputItem, null);
}
```

Výpis 3: Konvertovanie dát do konštant

V úlohe som využil svoje základné znalosti SQL získané v hodinách Databázových a informačních systémov. Naučil som sa vytvárať DataView a pochopil som komunikáciu medzi databázou, biznis vrstvou a web-klientskou vrstvou.

5.4 Odeslání objednávky e-mailem – OrderOutSendEmailActionCommandHandler

Táto úloha sa zaoberala pokynom odoslania e-mailu z myTEAM NG aplikácie. Na vypracovanie som dostal 24 hodín.

Prvým krokom pri tvorbe ActionCommandu je pochopenie, ako bude daný príkaz v kontexte fungovať. Musel som si uvedomiť, ktoré operácie predchádzali kliknutiu na tlačidlo odoslať a potom som mohol navrhnúť postup nasledujúcich operácií.

Užívateľ otvorí formulár a vyplní formulárové komponenty, minimálne všetky povinné, inak môže nastať prvá komplikácia. Je pravdepodobné, že bude nasledovať pridanie prílohy k objednávke, ktorá sa ale neuloží do databáze, lebo formulár ešte nebol stále odoslaný. Tu vznikla druhá komplikácia. Užívateľ nakoniec klikne na tlačidlo.

Druhý krok je založenie príkazu do správneho modulu, v tomto prípade do objednávok a následna implementácia v .NET kóde. Pokiaľ som chcel odoslať dáta z formuláru do databáze, musel som skontrolovať, či hodnoty prevzaté z konštant nastavených vo formulárových komponentách obsahujú validné dáta. Pokiaľ nie, vyhodí sa výnimka. To je zaistené vďaka override metódy predka s aktuálnými dátami a cieľovými tabulkami v databázi. Príloha je nepovinným atribútom. Ak ju e-mail obsahuje, som povinný skontrolovať, či je typ validný. Nesmie nastať možnosť odoslania víru alebo inak narušiť bezpečnosť a dôveryhodnosť systému. Komplikáciu s priložením ešte neexistujúcej prílohy som vyriešil vygenerovaním falošného, dočasného identifikátoru prílohy, ktorú som mohol pridať k záznamu e-mailu. Keď už som mal všetky validné dáta, mohol som ich preposlať službe na vytvorenie T4 šablóny, ktorá si ich následne nastavila ako atribúty a vygenerovala obsah šablóny. Následne som zavolal službu, ktorá sa stará o odosielanie e-mailov. V nej sa e-mail postavil do fronty, kde čakal na odoslanie. Po odoslaní sa uložil do databáze.

```
bool noAttachmentConfirmed=data.Value<bool>(
    OrderOutSendEmailActionCommandConstants.NoAttachmentConfirmedInputItem);
if (!noAttachmentConfirmed)
{
   Idx attachmentParentIdx=data.SafeGetValue<Idx>(
       OrderOutSendEmailDetailDefaultsDataViewConstants.
       _AttachmentPrimIdxOutputItem);
   int attachmentCount=GetAttachmentEntitiesCount(attachmentParentIdx);

   if (attachmentCount <= 0)
   {
      ValidationInfo validationResult = new ValidationInfo();
      validationResult.Items.Add(new ValidationInfoItem
      {
         Property=OrderOutSendEmailActionCommandConstants.
             NoAttachmentValidationErrorOutputItem,
         Text=OrderOutSendEmailActionCommandConstants.
             NoAttachmentValidationErrorOutputItem,
         Type=ValidationInfoItemType.Required
      });
      throw new ValidationInfoException(validationResult);
```

```
    }
}
Idx attachmentFakePrimIdx=data.SafeGetValue<Idx>(
    OrderOutSendEmailDetailDefaultsDataViewConstants.
    _AttachmentPrimIdxOutputItem);
```

Výpis 4: Validácia príloh

Úlohu som splnil v stanovenom termíne. Uplatnil som znalosti jazyka C# z hodín programovacích jazykov. Podarilo sa mi zlepšiť prácu s vývojovým prostredím Visual Studia vďaka využívaniu viacerých druhov služieb a typov konštant.

5.5 Zarovnávání notifikačních e-mailů

Obsahom úlohy bol prepis šablóny, generovanej voľne dostupným web builderom, do prehľadnejšieho, čitateľnejšieho a lepšie udržiavateľnejšieho formátu. V Agendách úloh, objednávok a faktúr mal užívateľ možnosť zaslania notifikačného e-mailu rôznym typom dodávateľom, objednávateľom, či zamestnancom podľa zvolenej úlohy, objednávky alebo faktúry. Pri vygenerovanej šablóne ale vznikol problém kompatibility s verziami e-mail prehliadačov (Google Chrome, Internet Explorer, Outlook), s ktorými pracoval klient. Na vypracovanie úlohy som dostal 24 hodín.

Na prácu so šablónami som použil Text Template Transformation Toolkit Framework. Jedným z hlavných dôvodov bola možnosť použitia kombinácie .NET kódu a HTML textových blokov, s ktorými mám skúsenosť z ASP.NET aplikácií. Dáta potrebné na namapovanie do premenných získam prostredníctvom odoslaných dát z formuláru na web-klientovi. Ako bolo spomenuté v predchádzajúcej úlohe, po spustení príkazu sa identifikátor, obsah e-mailu, doručovateľ a ostatné dáta pošlú na server, kde sa nastavia do atribútov riadiacej logiky šablóny. Tieto atribúty, funkcie, či výrazy je možné volať a vypísať použitím <#= #>.

Po zistení všetkých potrebných informácií a naštudovania materiálov som začal prácu so založením nového projektu pre tvorbu šablón. Voľba založenia nového projektu vyplývala z nemožnosti využitia HTML formátovania a zvýraznenia syntaxe, ktoré výrazne zlepšujú čitateľnosť a prácu s kódom. Ďalším faktorom bolo omeškanie odosielania e-mailov zo služby na myTEAM NG serveri (QAS), ktoré by spomalilo testovanie. Po vytvorení jednoduchej aplikácie na odosielanie e-mailov som začal s prípravou HTML šablóny. Bolo nutné si uvedomiť, že počet využitia šablón bude rásť s počtom vznikajúcich agend v aplikácií. Z toho vyplýva nutnosť skladania šablón z viacerých modulov, nech sa vyhneme zbytočnej opakovateľnosti kódu. To nám HTML a tým pádom T4 umožňuje. Po preskúmaní grafických návrhov bolo zjavné, že jediná identická a opakovane používaná časť kódu bude päta (footer) šablóny. Nasledovalo vytvorenie piatich šablón - päta, default a šablóny pre úlohy, objednávky, a faktúry. Všetky šablóny boli napísané bez štýlov v .scss súbore. Boli použité len základné štýly, aplikované v HTML kóde, z dôvodu využitia T4, jednoduchosti kódu a kompatibility s prehliadačmi. Všetky dáta boli dočasne nastavené staticky.

Posledným krokom bolo vloženie navrhnutých šablón na server QAS a aplikovanie T4 formátu. Dáta boli nahradené dynamickými hodnotami z atribútov riadiacej logiky T4 Template.

Úlohu som splnil do 16 hodín. Naučil som sa využívať T4 Template a lepšie som porozumel znovupoužiteľnosti kódu. Precvičil som HTML a C# programovací jazyk.

Obr. 5: Výsledná šablóna notifikačného e-mailu úlohy

5.6 WebClient – ValidationErrorsBlock Component

Úloha na ValidationErrorsBlock bola mojou prvou úlohou na vytvorenie jednej zo základných komponent. Základné komponenty sa využívajú v prevažnej časti aplikácie, po prípade z nich dedia ďalšie komponenty. Dôvody vzniku komponenty boli dva. Návrh analytikov v spolupráci s klientom, ktorý požadoval grafické vyzobrazenie validačných hlášok prispôsobených designu aplikácie. A tento návrh prišiel v čase, keď sme chceli rozdeliť validačné hlášky web-klienta a serveru. Na vypracovanie úlohy mi bol pridelený čas 40 hodín.

Mojim prvým krokom bolo pochopenie rozdelenia validačných hlášok. Pokiaľ nastane chyba v aplikácií alebo v databáze, tak server pošle na web-klienta pokyn s chybovou hláškou, ktorá sa

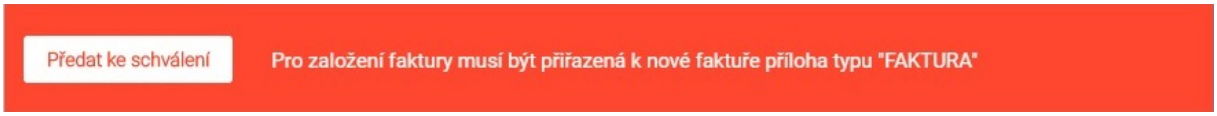
zobrazí v modálnom okne. Týmto spôsobom to fungovalo doteraz pre všetky hlásenia chýb. Toto riešenie nebolo vhodné pre užívateľa, lebo chybovú hlášku z databáze nepochopí a nemá spôsob, ako ju vyriešiť, oproti zvyčajnej validačnej chybe. Po rozdelení, by mal užívateľ vyplniť formulár. Ak užívateľ nemá práva alebo nastane chyba, po prípade, zabudne pridať prílohu u zakladania novej faktúry a stlačí tlačidlo odoslať, príkaz sa pokúsi odoslať dáta cez server do databáze. Na strane serveru ale príkaz neprejde cez validovanie, vyhodí sa výnimka a odošle sa v podobe dát na web-klienta. Na strane web-klienta som dáta spracoval prostredníctvom zdedenej premennej, kde sa uložili. S touto premennou som mohol ďalej pracovať. Vyfiltroval si z dát validačné hlášky a uložil ich obsah do poľa. Takto spracované pole môžem data-bindingom vypísať v HTML. Ako najlepšiu možnosť spracovania poľa a výpis jeho prvkov v HTML som zvolil *ngFor direktívu. Jej úlohou je opakovať daný prvok DOM (alebo kolekciu DOM prvkov) zakaždým, pre inú hodnotu v poli. Podobná funkcionalita ako foreach. Túto triedu bloku s chybovými hláškami som zabalil do triedy s direktívou *ngIf, ktorá sa používa, keď chceme zobraziť alebo skryť prvok na základe podmienky. Podmienka je určená výsledkom výrazu, ktorý vložím do direktívy. Vložený výraz skontroloval, či pole nie je prázdne a zároveň, či je jeho veľkosť väčšia než nula.

Posledným krokom bolo nastavenie štýlov pre spomenuté triedy v CSS. Tie som navolil podľa prezentácie vytvorenej grafikom. Pri zobrazení validačnej chyby v pätičke sa mala zároveň aj prefarbiť. Túto funkčnosť som nastavil zavolaním novej triedy v prípade, že panel v premennej obsahuje chybu.

```
<div class="row" [class.red]="hasError">
   <div class="column no-padding">
      <ng-content select="panel-buttons"></ng-content>
   </div>
   <div class="column no-padding" *ngIf="visible && errorList != null &&
       errorList.length > 0">
      <div class="errorBlock">
         <span class="value error"*ngFor="let error of errorList">{{error.Text
             || error}}</span>
      </div>
   </div>
</div>
```

Výpis 5: ValidationErrorsBlock komponenta v jazyku HTML

Úlohu som splnil v stanovenom termíne. V tomto prípade sa komponenta začala využívať v päte každého panelu. Počas refactoringu som sa musel ku komponente vrátiť a prispôsobiť ju novým návrhom grafika. Vďaka úlohe som sa naučil písať kód efektívnejšie, s nadhľadom na opätovné využitie v rôznych prostrediach aplikácie a osvojil si používanie direktív.



Obr. 6: Výsledné zobrazenie validačnej chyby ValidationErrorsBlock komponenty

5.7 WebClient - Podnadpisy - State/ControlData Extension

Funkčnosť obsahu WorkItemu v aplikácií myTEAM NG zabezpečujú Extenzie. Extenzie sú objekty obsahujúce metódy, ktoré sa starajú o inicializáciu, spracovanie a predávanie dát na web-klientovi. Tým pádom sú písané programovacím jazykom TypeScript. Jednou z mnoha úloh Extenzií je aj nastavenie podnadpisu panelu. Podnadpis sa môže získať dvomi spôsobmi:

1. Načítať ako súčasť dotazu z databáze
2. Získať ho z atribútu stavu predchádzajúceho panelu, ktorý je zároveň predkom

Tieto dve možnosti sú naimplementované. Mojou úlohou bolo vytvoriť Extenziu, ktorá by zahrňovala funkčnosť oboh variant nastavovania podnadpisu. Vytvorenie takého typu Extenzie by uľahčil prácu pri tvorbe ďalších panelov. Programátori by nemuseli premýšlať, ktorá Extenzia je vhodnejšia pre daný typ panelu. Úlohu som splnil v termíne 8 hodín.

Na začiatok som musel naštudovať a pochopiť implementáciu Extenzií, z ktorých som vychádzal. Po preštudovaní metód som usúdil, že funkčnosť oboch Extenzií nebolo potrebné zrefaktorovať a mohla ostať zachovaná. Extenzia východzajúca z dát sa aktualizovala pri zmene hodnoty dát. Orientovala sa podľa zmeny identifikátoru vybranej položky, ktorá sa nastavila ako prvý vstupný parameter. Podľa identifikátoru a dotazu, vloženého ako druhý vstupný parameter, sa načítala hodnota z dát do podnadpisu panelu. Hlavnou nevýhodou tohto prístupu je zbytočnosť posielania celého dotazu na server a následne na databázu, pokiaľ jediné potrebné dáta pre panel sú na podnadpis. V týchto prípadoch je lepšia varianta získavania podnadpisu z atribútu stavu predka. V Extenzií východzajúcej zo získavania stavu sa podnadpis nastavuje z rekurzívnej funkcie, ktorá prechádza predka za predkom, pokiaľ nenájde nenulový atribút stavu. Z týchto informácií som usúdil, že najvhodnejšou variantou je implementácia podmienky, v ktorej nastavím premennú podnadpisu prostredníctvom stavu a pokiaľ sa funkcií nepodarí nájsť vhodný stav predka, tak sa obrátim na dotaz odkazujúci sa na databázu.

Následne som Extenziu rozšíril o vstupný parameter prefixu, ktorý sa ako reťazec, pridá k reťazcu podnadpisu. To umožní programátorovi najjednoduchšiu cestu k doplneniu celkového podnadpisu. Na konci musím spomenúť, že implementácia pôvodných Extenzií zostala ponechaná v aplikácií a využíva sa pri špeciálnych prípadoch.

```
var subtitle: string = this.stateKey;
var prefixText: string = this.prefixText;
if (subtitle)
```

```
{
   if (prefixText && !subtitle.startsWith(prefixText))
   {
      subtitle = prefixText + " " + subtitle;
   }
   workItem.subtitle = subtitle;
   workItem.workItemComponent.markAndDetectChanges();
}
else
{
   var controlData: ControlData = this.controlData;
   if (!inConnector || !controlData)
   {
      return;
   }
   var keyObject: Object = inConnector.value;
   if (keyObject == null)
      return;
   var parameters = {};
   parameters[this.parameterName] = <string>keyObject;
   controlData.execute(parameters);
}
```

Výpis 6: Podmienka s nastavením podnadpisu v jazyku TypeScript

Z úlohy som si odniesol prvú skúsenosť s prácou na Extenziách, rozšírenie znalostí jazyku TypeScript a lepšie pochopenie funkčnosti WorkItemov.

5.8 Objednávky vydané - Zobrazení názvu stavu objednávky

Vyriešenie tejto úloha mi bolo pridelené na základe chyby, ktorá bola zapísaná do registru chýb testermi.

Po preštudovaní analýzy a kooperácií s analytikmi sme narazili na nekonzistentnú analýzu s databázou. Podľa popisu v analýze, mal byť vyzobrazený atribút kód stavu objednávky v podobe pomenovania stavu. V databáze ale kód obsahoval číselnú hodnotu stavu a pomenovanie daná tabuľka neobsahovala. Zobrali sme teda do úvahy viacero spôsobov riešenia problému. Ako aj úpravu databáze či nastavovanie hodnôt na serveri. Dohodli sme sa, že najlepšie bude odoslať hodnotu v podobe kódu na web-klienta a tam chybu vyriešiť.

Zvolil som podľa môjho názoru najjednoduchšie riešenie. Pridal som na web-klientovi Helper, ako menšiu službu, ktorý sa bude starať o preklad kódu stavu na názov. Helper, sa teda skladá z funkcie obsahujúcej prepínač switch. Ako vstupný parameter funkcie sa predáva kód

stavu objednávky a ten sa následne predá prepínaču. Potom som implementoval enum, obsahujúci všetky typy stavou objednávky. Switch, teda porovnáva hodnotu na vstupe s hodnotami v enume. Pokiaľ narazí na zhodu, funkcia vráti hodnotu v podobe reťazca nadefinovanú v XML.

```
switch (state)
{
      case OrderOutState.Concept:
            return this.stringTableService.getText("OrderOut.State.Concept");
      case OrderOutState.ToppingTheDials:
            return this.stringTableService.getText("OrderOut.State.
               ToppingTheDials");
      case OrderOutState.Approval:
            return this.stringTableService.getText("OrderOut.State.Approval");
      case OrderOutState.Denied:
            return this.stringTableService.getText("OrderOut.State.Denied");
      case OrderOutState.SendingToSupplier:
            return this.stringTableService.getText("OrderOut.State.
               SendingToSupplier");
      case OrderOutState.AuthenticationBySupplier:
            return this.stringTableService.getText("OrderOut.State.
               AuthenticationBySupplier");
      case OrderOutState.RealisationBySupplier:
            return this.stringTableService.getText("OrderOut.State.
               RealisationBySupplier");
      case OrderOutState.Delivered:
            return this.stringTableService.getText("OrderOut.State.Delivered");
      case OrderOutState.DeliveredPartially:
            return this.stringTableService.getText("OrderOut.State.
               DeliveredPartially");
      case OrderOutState.HandlingObjections:
            return this.stringTableService.getText("OrderOut.State.
               HandlingObjections");
      case OrderOutState.Canceled:
            return this.stringTableService.getText("OrderOut.State.Canceled");
      case OrderOutState.RealizeAdHoc:
            return this.stringTableService.getText("OrderOut.State.RealizeAdHoc"
               );
      default:
            return "";
```

```
}
```

Výpis 7: Switch v OrderOutHelper

Potom už len stačilo naimportovať Helper do danej komponenty. Vytvoriť funkciu, ktorou predám kód objednávky Helpru a ten mi hodnotu vráti na výstup. Následne data-bindingom zavolať v HTML kóde funkciu, ktorá konečne vráti požadovaný reťazec.

6 Zhodnotenie bakalárskej praxe

6.1 Uplatnenie znalostí nadobudnutých v škole

Pri výkonávaní praxe som využil znalosti objektovo-orientovaného programovania z hodín Programování 2 a Programovací jazyky 1, 2. Tiež som využil vedomostí z Alogoritmov 2. Neoceniteľnou oporou mi boli nadobudnuté znalosti jazyka C#, SQL dotazov a T-SQL príkazov z predmetov Úvod do databázových systémov, Databázové a informačné systémy a Vývoj informačných systémov. Taktiež hodiny Softwarového inžinierstva mi pomohli porozumieť procesu vývoja aplikácie.

6.2 Chýbajúce znalosti

Mrzia ma chýbajúce znalosti z voliteľného predmetu Vývoj internetových aplikácií, ktorý som do osobného štúdijného plánu nezahrnul. Verím, že dostupné informácie z tohoto predmetu by som počas praxe využil. Všetky chýbajúce znalosti som samoštúdiom doplnil, aj za pomoci materiálov zo spomínaného predmetu. Do budúcna by som ocenil, keby sa v škole vyskytol predmet so zameraním na webové technológie a frameworky, hlavne na tie najnovšie, ako AngularJS.

6.3 Získané znalosti

Určite k získaným znalostiam musím priradiť znalosť frameworku Angular 2 a jazyka TypeScript, v ktorých chcem do budúcna pokračovať. Kladne hodnotím zlepšenú znalost jazyka C# a SQL. Je pre mňa veľkým plus z efektívnenie využívania Visual Studia a Microsoft SQL Management Studia.

7 Záver

Začiatok bakalárskej praxe a zorientovanie sa v novom, profesionálnom prostredí bolo ťažké. Za jeden zo svojich mála úspechov považujem, že sa mi po necelom prvom týždni podarilo zapojiť do vývoja aplikácie a vývojového procesu. Moja spolupráca so spoločnosťou KVADOS, a.s. bola intenzívnejšia a nad rámec povinne odpracovaných hodín bakalárskej praxe. Aj vďaka tomu a radám mentora sa mi podarilo bez väčších problémov začleniť do vývojového tímu. Som rád, že som svojou prácou bol prospešný a prispel k splneniu všetkých deadlinov zadaných klientom.

Bakalársku prax hodnotím pozitívne. Z praxe som si odniesol veľa skúseností z hladiska programovania, Workflow a práce v týme. Ranné meetingy tiež hodnotím kladne, bolo očividné, že prinášajú výsledky a veľa som si z nich odniesol. Spolupráca sa niesla v príjemnom kolegiálnom prostredí. Účel praxe bol zaškoliť ma a pripraviť na prácu software developera. Myslím, že tento cieľ bol splnený. Taktiež mi spoločnosť KVADOS ponúkla zamestnaneckú zmluvu na plný úväzok. Vedenie bolo ústretové a bez problémov sme sa dohodli na skrátenom úväzku kvôli môjmu pokračovaniu v štúdiu.

Budem sa tešiť na ďalšiu spoluprácu so spoločnosťou KVADOS a som vďačný za možnosť absolvovania bakalárskej praxe.

Literatúra

[1] Nate Murray, Felipe Coury, Ari Lerner, Carlos Taborda. *Ng-book 2.*. San Francisco, California: FULLSTACK.io, 2017. ISBN 978-099-1344-611.

[2] *Team Fountadation Server* [online][cit.2017-04-19]. Dostupné z: `https://www.visualstudio.com/en-us/products/tfs-overview-vs.aspx`.

[3] *Writing a T4 Text Template* [online][cit.2017-04-19]. Dostupné z `https://msdn.microsoft.com/en-us/library/bb126478.aspx`.

[4] *Single-Page Application* [online][cit.2017-04-19]. Dostupné z `https://msdn.microsoft.com/en-us/magazine/dn463786.aspx`.

[5] *Command and Query Responsibility Segregation* [online][cit.2017-04-19]. Dostupné z `https://docs.microsoft.com/en-us/azure/architecture/patterns/cqrs`.

[6] *Informace o společnosti KVADOS* [online][cit.2017-04-19]. Dostupné z `https://www.kvados.cz/o-spolecnosti/`.

A Príloha

Obsah priloženého CD

- Elektronická verzia Bakalárskej práce